

プログラミング演習B

ML編 第2回

2009/4/21 (コミ)

2009/4/22 (情報・知能)

住井

[http://www.kb.ecei.tohoku.ac.jp/
~sumii/class/proenb2009/ml2/](http://www.kb.ecei.tohoku.ac.jp/~sumii/class/proenb2009/ml2/)

今日のポイント

1. 変数定義

val 変数名 = 式

2. 関数定義

fun 関数名 引数名₁ ... 引数名_n = 式

3. 単一代入と静的スコープ

4. 再帰関数

レポートについて

電気・情報系内のマシンから

<http://130.34.188.208/> (情報・知能)

<http://130.34.188.209/> (コミ)

にアクセスし、画面にしたがって提出せよ。締め切りは**一週間後厳守**。

- 初回は画面にしたがい自分のアカウントを作成すること。
- 「プログラム」のテキストボックスがある課題では、プログラムとしてsmlに**入力**した文字列のみを**過不足なく正確に**コピー&ペーストして提出せよ。
(smlの**出力**は「プログラム」ではなく考察に含めて書くこと。)
- プログラムの課題でも必ず考察を書くこと。
- 提出したレポートやプログラムの実行結果は「提出状況」から確認できる。
 - 質問はm1-enshu@kb.ecei.tohoku.ac.jpにメールせよ。
 - レポートの不正は試験の不正と同様に処置する。

ポイント 1 : 変数定義

プログラムに繰り返し出てくる
「同じ式」を一つにまとめたい
⇒ 「変数」を定義する

例題：

1. 半径1.2の円の周の長さを求めよ。
2. 直径3.4の円の周の長さを求めよ。
3. 半径5.6の円の面積を求めよ。

ただし円周率は3.14159265359とする。

変数を定義しないと...

- 2.0 * 3.14159265359 * 1.2 ;

val it = 7.53982236862 : real

- 3.14159265359 * 3.4 ;

val it = 10.6814150222 : real

- 3.14159265359 * 5.6 * 5.6 ;

val it = 98.5203456166 : real

変数を定義すれば...

- `val pi = 3.14159265359 ;`

`val pi = 3.14159265359 : real`

- `2.0 * pi * 1.2 ;`

`val it = 7.53982236862 : real`

- `pi * 3.4 ;`

`val it = 10.6814150222 : real`

- `pi * 5.6 * 5.6 ;`

`val it = 98.5203456166 : real`

変数定義の構文

val 変数名 = 式

「**変数名**」は英文字で始まり、英数字
または_（下線）または'（アポスト
ロフィ）が続く

- 大文字も小文字も使用できるが区別される
- 実は!#\$%など記号列も良いが本演習では使わない

ちなみに...

今まで**式;**と入力していたのは、
実は**val it = 式;**の省略

- **2 ;**

val it = 2 : int

- **it * 2;**

val it = 4 : int

- **it * 2;**

val it = 8 : int

- **it * 2;**

val it = 16 : int

課題 2.1

現在の円ドル為替レートを調べ、
「1ドル＝何円か」を表す変数
rateを定義して、次の計算をせよ。

1. 123.45ドルは何円か。
2. 6789円は何ドルか。
 - 結果は小数のままでよい。
 - 買いレートと売りレートの差など、細かいことは気にしなくて良い。

ポイント2：関数定義

繰り返し出てくる「同じ形の式」
を一つにまとめたい

⇒「関数」を定義する

例題：

1. 半径1.2の円の面積を求めよ。
2. 半径3.45の円の面積を求めよ。
3. 半径6.789の円の面積を求めよ。

関数を定義すれば簡単

```
- val pi = 3.14159265359 ;
```

```
val pi = 3.14159265359 : real
```

```
- fun area r = pi * r * r ;
```

```
val area = fn : real -> real
```

```
- area 1.2 ;
```

```
val it = 4.52389342117 : real
```

```
- area 3.45 ;
```

```
val it = 37.3928065594 : real
```

```
- area 6.789 ;
```

```
val it = 144.797642174 : real
```

関数定義の構文

**fun 関数名 引数名₁ ... 引数名_n =
式**

- 改行はどこに入れてもいいが、
=の後に入れるのが普通
- 関数名・引数名に使える文字列は
変数名と同じ

関数適用の構文

関数 引数₁ ... 引数_n

- 関数と引数を並べて書くだけ

注：関数適用の優先順位は二項演算より高い

例：f 3 + g 4は(f 3)+(g 4)と同じ

- 関数に引数を与えて呼び出すことを、
「関数を引数に適用する」という
 - 「引数を関数に適用する」とは言わないので気をつける

課題 2. 2

次の関数を定義し、
それらを適用した例を挙げよ。

1. 整数 i を受け取って、
 $i+1$ を返す関数 `succ`
2. 整数 i を受け取って、
 $i*i$ を返す関数 `square`
3. 浮動小数点数 x を受け取って、
 $x/2.0$ を返す関数 `half`

課題 2. 3

課題 2. 1 で定義した変数 `rate` を用いて、次の関数を定義せよ。

1. 円をドルに換算する関数
`yen_to_usd`
2. ドルを円に換算する関数
`usd_to_yen`

課題 2.4

浮動小数点数 x と y を受け取って、座標平面における原点から点 (x, y) までの距離を返す関数`distance`を定義せよ。

- 平方根を計算する関数`Math.sqrt`はあらかじめ定義されているので用いて良い。

ヒント：次のようになれば良い。

```
- distance 3.0 4.0 ;  
val it = 5.0 : real
```

ポイント3

下の式の評価結果はいくつになるか？

```
- val pi = 3.14 ;
```

```
val pi = 3.14 : real
```

```
- fun area r = r * r * pi ;
```

```
val area = fn : real -> real
```

```
- val pi = 3.0 ;
```

```
val pi = 3.0 : real
```

```
- area 10.0 ;
```

```
val it = ? ? ? ? ? : real
```

別の例

- **val x = 123 ;**

val x = 123 : int

- **x = x + 1 ;**

val it = false : bool

- **x ;**

val it = 123 : int

なんでそうなるの？

- Cなどの命令型言語と異なり、関数型言語MLでは**変数の値が定義の後に変化することはない（単一代入）**。
- たとえ同じ名前を定義しても、それは新しい変数の定義であって、それより前の定義には**影響しない（静的スコープ）**。

ポイント4：再帰関数

例題：

正の整数 n を受け取って、
1から n までの整数の総和を返す
関数`sum`を定義せよ。

考え方のコツ

nについての**場合分け**と**帰納法**

1. nが1の場合：
1を返す
2. nが1より大きい場合：
1からn-1までの総和である
 $\text{sum}(n-1)$ を求め、
それにnを足して返す

できたプログラム

```
- fun sum n =  
=   if n = 1 then 1 else  
=   sum (n - 1) + n ;  
val sum = fn : int -> int  
- sum 10 ;  
val it = 55 : int
```

ただし「if 式₁ then 式₂ else 式₃」は
「式₁の値がtrueならば式₂の値を、
falseならば式₃の値を返す」という式

課題 2.5

非負整数 n を受け取り、フィボナッチ数列の第 n 項を計算する関数`fib`を、次の考え方に基づいて定義せよ。

1. n が1以下ならば n を返す
2. そうでなければ、第 $n-1$ 項である`fib( $n-1$ )`と、第 $n-2$ 項である`fib( $n-2$ )`との和を返す

課題 2. 6

浮動小数点数 x と非負整数 n を受け取り、 x の n 乗を返す関数`power`を、次の考え方に基づいて定義せよ。

1. n が0ならば1.0を返す
2. そうでなければ、 x の $n-1$ 乗である`power x (n-1)`を求め、それに x を掛けて返す

課題 2.7

二つの非負整数 m と n を受け取り、 m と n の最大公約数を返す関数 gcd を、次の考え方に基づいて定義せよ。

1. m が0ならば n を返す
2. そうでなければ、もし m が n 以下だったら、 m と $n-m$ の最大公約数である $\text{gcd } m \ (n-m)$ を返す
3. そうでなければ、 n と $m-n$ の最大公約数である $\text{gcd } n \ (m-n)$ を返す

課題 2.8 (optional)

※ optional: やらなくても良いが、出来たらボーナス点

1. アッカーマン関数とは、どのような関数か。検索などで調べて述べよ。
2. SMLでアッカーマン関数を定義し、関数の特徴を実際に確認せよ。

課題 2.9 (optional)

※ optional: やらなくても良いが、出来たらボーナス点

1. n が偶数の場合は

x の n 乗 = $(x * x)$ の $(n \text{ div } 2)$ 乗

であることを利用して、課題 2.6 の関数 `power` を「大幅に」高速にせよ (`power 0.1 10000000000` が一瞬で計算できるぐらい)。

2. 高速化後の `power` を呼び出すごとに、浮動小数点数の掛け算は何回ぐらい計算されるか。 n の関数 (ML の関数ではなく数学の関数) として大まかに表せ。
3. 1. と同様の関数を、C 言語ないし Java 言語で、再帰ではなくループを用いて書け。